1992-02

On the design of cost-tables for realizing multiple-valued circuits

Schueller, Kriss A.

On the design of cost-tables for realizing multiple-valued circuits," IEEE Transactions on Computers, Vol. 41, No. 2, February 1992, pp. 178-189

On the Design of Cost-Tables for Realizing Multiple-Valued Circuits

Kriss A. Schueller, *Member, IEEE*, and Jon T. Butler, *Fellow, IEEE*

Abstract— **We propose a heuristic for finding minimal cost-tables for use in the design of multiple-valued logic circuits. It is an iterative approach, in which a good table of size t is composed of a good table of size $t-1$, etc. We analyze its performance, comparing it with three other heuristics. The importance of finding good cost-tables is demonstrated by an analysis that shows there is a wide variation in both cost-table performance and in the performance of heuristics for generating cost-tables.**

We study linear cost, a general cost function of which two previously studied cost functions are special cases. It is shown that the minimal cost-table using one of the (infinitely many) linear cost functions is identical to a minimal cost-table using *any* other linear cost function. Thus, a heuristic for finding the minimal cost-table using the linear cost function is independent of the specific cost function parameters. This result and our observation of well-studied nonlinear cost functions indicate that cost-table design is only marginally dependent on the cost function.

We show two additional results on cost-table design. First, it is demonstrated that a search for minimal cost-tables cannot exclude certain seemingly useless functions called composite functions. Second, while the complexity of cost-table design appears to preclude a computationally efficient general algorithm for finding the minimal cost-table, a special case allows efficient design. For the case of a small cost-table, we show how to find the minimal cost-table.

Index Terms—**Cost-table, logic design, minimization, multiple-valued logic, synthesis.**

I. Introduction

In the classical synthesis of logic functions, a given function is realized as a set of component functions that are combined by a connecting function. For example, in binary minimal sum-of-products synthesis, the component functions are the AND of variables or their complements and the connecting function is the OR. Determining which component and connecting functions to provide the designer has been traditionally an ad hoc process, depending on the perceived usefulness and cost of supplied functions. Cost is an especially important factor, and is determined by the technology used. For example, in multiple-valued charge-coupled device (CCD) logic [6], the sum operator is especially inexpensive, and so it occurs frequently in realizations. However, the value of a component or connective function depends also on the extent to which it can be used to realize other functions. There has been little formal study of this problem.

A formalization of this process is design by cost-table. In the cost-table approach, components are chosen from a table and combined to fulfill the design specifications at the least cost. Cost is the sum of the costs of the components plus the cost of combining them. The use of cost-tables is universal. For example, the writing of a program is essentially a design by cost-table. Here, entries are instructions and cost can be execution speed. Similarly, VLSI layout is a cost-table approach where the table is a library of modules and cost can be chip area. The question of reduced-instruction set computers verses complex-instruction set computers is a matter of whether low cost (simple) instructions or high cost (complex) instructions should be provided in a cost-table consisting of machine instructions.

The need for design techniques for CCD multiple-valued logic circuits [6], [7] has inspired interest in the cost-table approach [1]–[3], [5], [7], [8], [10], [14]. Here, cost represents chip area, power dissipation, speed, etc. Given a function, there may be many ways to realize it using cost-table functions, and we are interested in one with lowest cost. This is called the *cost-table realization problem.*

The concept of a cost of realizations has long been a part of the study of multiple-valued logic. For example, Allen and Givone [4], Miller and Muzio [9], and Smith [15] have used cost measures in evaluating sum-of-products expressions. The first use of the *cost-table* in the design of multiple-valued logic circuits was by Kerkhoff and Robroek [7] and Robroek [11]. Their proposed table contains 45 functions, from which all 256 one-variable functions are synthesized. The cost of each function in the cost-table is an approximation to the chip area occupied by a CCD realization of that function. The synthesis technique used is exhaustive search. Lee and Butler [8] show a cost-table of 24 entries that produces realizations as good as or better than those in [7] and [11]. The proposed synthesis algorithm is still a search; however, nonproductive combinations are eliminated by using the transition count of the function to guide the search. Abd-El Barr, Vranesic, and Zaky [2] propose two heuristics for implementing one- and two-variable functions. For one-variable functions, the design uses the break count of a given function, and this results in an improvement in the realization of 20% of the one-variable functions considered in [8]. Abd-El Barr, Vranesic, and Zaky [3] analyze the realizations of one-variable functions used in the cost-table technique. By an

Manuscript received April 16, 1987; revised February 12, 1991. This work was supported by NATO Grant 423/84, by NSF Grant MIP-8706553, and by NRL under an NPS direct-funded grant.

K. A. Schueller is with the Department of Mathematics and Computer Science, Youngstown State University, Youngstown, OH 44555.

J. T. Butler is with the Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA 93943.

IEEE Log Number 9103108.

enumerative process, a cost reduction is achieved in 76% of the functions.

Unlike the cost-table realization problem, the *minimal cost-table problem* concerns the design of a cost-table, i.e., selecting the functions in the table. Here, the choice of a cost-table is determined by the average cost of the realizations produced; for a given cost-table size, one wants a cost-table that yields the lowest average cost. Schueller, Tirumalai, and Butler [13] show a heuristic for finding a cost-table that is minimal or nearly so, and from this, it was found that the cost-tables of [7] and [8] are not minimal. For 85% of the cost-table sizes over which it was applied, the resulting tables are *provably* minimal. No proof of minimality exists for the remaining 15%. Most are believed to be minimal, although for one size, the generated cost-table is provably not minimal. These results show that there is a point of diminishing returns with respect to cost-table size. That is, while cost-tables of larger size produce more economical realizations, beyond a certain size, about 10% of the total number of functions to be synthesized, there is little benefit to adding more functions to the cost-table. The analysis in [13] was done for five different costs, and it was found that the point of diminishing returns is approximately the same for all costs.

In this paper, we analyze the minimal cost-table problem. In addition, to the heuristic in [13], we consider three others. All four heuristics are analyzed over cost-tables on the set of one-variable 4-valued functions, one of the few sets where such an analysis is computationally possible. It is shown that one heuristic (the one given in [13]) is significantly better that the others. Additionally, we generate a set of random cost-tables with specific sizes and compare the average total cost with heuristically generated best total costs. For the case where cost-tables are small relative to the number of functions realized (including practical cost-tables on multiple-valued functions of two or more functions), we find that it is important to carefully choose the heuristic. On the other hand, for larger sizes, we find that almost *any* cost-table gives good results.

As in [13], our analysis covers five cost functions. We show that two of these, which appear to be different, are really members of a single class, called linear cost functions, and that their characteristics are amenable to an analysis. For example, we show how to identify all minimal cost-tables of size 1 larger than the smallest size. Also, we show that the minimal cost-table of a given size is the same for any of the infinitely many linear cost functions. That is, if the existing cost function is linear, the resulting minimal cost-table of some specified size is independent of the particular linear parameters used. This formal statement on a set of specific cost functions agrees with our observation about other types of cost functions; *cost functions seem to have a marginal effect on the composition of the minimal cost-table.*

We analyze the composition of functions that belong to the minimal cost-table. Specifically, we show that a search for a minimal cost-table cannot exclude certain functions, called composite functions, that are best realized as a combination of other functions. This is a surprising result, since, in large cost-tables, composite functions are unnecessary. They do not contribute to any function realization.

II. Background and Notation

Let $R = \{0, 1, \cdots, r-1\}$ be a set of r logic values, where $r \geq 2$. Let $X = \{x_1, x_2, \cdots, x_n\}$ be a set of n variables, where x_i takes on values from R. A function $f(X)$ is a mapping $f : R^n \rightarrow R$. If X is a single variable x, $f(x)$ is represented as an r-tuple, $\langle f(0), f(1), \cdots, f(r-1)\rangle$. For example, if $r = 4$, then $f(x) = \langle 3, 2, 1, 0\rangle$ represents a *complement* function in which 0 maps to 3, 1 to 2, 2 to 3, and 3 to 0. Let $U_{n,r}$ be the set of all r-valued functions of n r-valued variables. Let $c(f)$, the *cost* of function f, be a mapping $c : U_{n,r} \rightarrow \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers. The cost function $c(f)$ introduced by Kerkhoff and Robroek [7], [11] for the design of 4-valued CCD logic circuits correlates closely with the chip area occupied by the most compact implementation of f. The cost function can also be chosen to correlate with other quantities like speed and power dissipation, allowing a range of parameters to be optimized.

In the realization of a given function by cost-table, cost-table functions are combined using a connecting operation. The connecting operation + between functions used in this paper is similar to ordinary addition with logic values viewed as integers. That is, if $f(X)$ is represented as the sum $f(X) = f_1(X)+f_2(X)+\cdots+f_m(X)$, then, for any assignment $\boldsymbol{v}$ of values to X, $f(\boldsymbol{v}) = f_1(\boldsymbol{v}) + f_2(\boldsymbol{v}) + \ldots + f_m(\boldsymbol{v})$, where each $f_i(\boldsymbol{v})$ is taken as an integer and where + is ordinary addition, except when the sum exceeds $r-1$, the highest output logic value, in which case + is undefined. For example, if $f_1(x) = \langle 0, 1, 2, 3\rangle$ and $f_2(x) = \langle 3, 2, 1, 0\rangle$, then $f_1(x) + f_2(x) = \langle 3, 3, 3, 3\rangle$ and $f_1(x)+f_1(x)$ is undefined. The first example shows that the sum of the identity function $\langle$0,1,2,3$\rangle$ and the complement function $\langle$3,2,1,0$\rangle$ is the constant function $\langle$3,3,3,3$\rangle$.

Let s be the cost of realizing the sum operation (+) between two functions. Thus, the cost of the realization $f = f_1+f_2+\cdots+f_m$ is $c(f_1) + c(f_2) + \cdots + c(f_m) + (m-1)s$, where the last term is the cost of $(m-1)$ two-input adders.

Function f is a *basis function* iff $f(X)$ is 1 for exactly one assignment of values to X and is 0 otherwise. Let BT be the set of all basis functions plus the constant 0 function (i.e., for all assignments of values to x, the constant 0 function is 0). A set of functions F is a *cost-table* iff $BT \subseteq F \subseteq U_{n,r}$. The condition $BT \subseteq F$ guarantees that all functions can be realized as the sum + of cost-table functions. For example, in $U_{1,4}$, $BT = \{\langle$0,0,0,0$\rangle$, $\langle$0,0,0,1$\rangle$, $\langle$0,0,1,0$\rangle$, $\langle$0,1,0,0$\rangle$, $\langle$1,0,0,0$\rangle\}$. If $\langle$0,0,0,1$\rangle$ is missing, it is impossible to realize certain functions, including $\langle$0,0,0,1$\rangle$ itself. Conversely, any function $\langle a_0, a_1, a_2, a_3\rangle$ can be realized as the sum of functions exclusively from BT, specifically a_0 functions of the form $\langle$1,0,0,0$\rangle$, a_1 $\langle$0,1,0,0$\rangle$, a_2 $\langle$0,0,1,0$\rangle$, and a_3 $\langle$1,0,0,0$\rangle$. BT is called the *basis cost-table.*

$c_F(f)$, the cost of realizing $f \in U_{n,r}$ with respect to cost-table F is the cost of the minimal cost realization, specifically

$$c_F(f) = \min_{f_1, f_2, \cdots, f_m \in F} \{c(f_1)+c(f_2)+\cdots+c(f_m)+(m-1)s\}$$

where $f = f_1+f_2+\cdots+f_m$ and c is a cost function. $f = f_1+f_2+\cdots+f_m$ is said to be a *minimal realization* of f, if there is no other realization of lower cost. The *total cost*, $T(F)$,

of cost-table F is

$$T(F) = \sum_{f \in U_{n,r}} c_F(f).$$

That is, $T(F)$ is the sum of the minimal costs of all functions realized under F. F_t is a *minimal cost-table* of size t iff $|F_t| = t$ and $T(F_t) \leq T(F)$, for all F such that $|F| = t$. A *near-minimal* cost-table is a cost-table with a total cost that is close to minimal and may even be minimal but has not been proven so. Dividing $T(F)$ by $|U_{n,r}|$ yields the *average cost* of realizing a function by cost-table F. Thus, if F is a minimal cost-table, then the average cost of realizing functions with F is no greater than with any other cost-table of the same size. If any function is as likely to be realized by the cost-table method as any other function, then a suitable criteria for judging cost-tables is average cost, or equivalently, total cost. Thus, for any given size t, we seek a cost-table with the smallest total cost. This is the *minimal cost-table problem.*

To assess the importance of finding a minimal cost-table, it is necessary to see the range of cost-table quality. F_t is a *maximal cost-table* of size t if $|F_t| = t$ and $T(F_t) \geq T(F)$, for all F, such that $|F| = t$. In the analysis to follow, representatives of maximal cost-tables will be used to show the range of cost-tables available. A *near-maximal* cost-table is a cost-table with a total cost that is close to maximal and may even be maximal but has not been proven so.

Since there is only one basis cost-table of size $r^n + 1$, it is minimal. However, the task of finding a minimal cost-table of other sizes is typically nontrivial. For no known nontrivial cases has the minimal cost-table problem been previously solved. In Section VII of this paper, we show a nontrivial case where this problem can be solved. However, an efficient process for finding a minimal t-entry cost-table F_t is needed that applies to *all* cases. We compare four heuristics for finding near-minimal cost-tables, all of which are based on the premise that a minimal cost-table of size t is likely to contain a minimal cost-table of size $t - 1$. To compare these heuristics, we consider one-variable 4-valued functions of which there are 256. For 4-valued functions with more than one variable, it is impossible to compare average or total costs of cost-tables. For example, to compute the average or total cost of a cost-table on two-variable 4-valued functions, it is necessary to compute the costs of $4^{4^2} \approx 4 \times 10^9$ functions! Since we are interested in the cost-table approach in general, our analysis is done with five different cost functions. They are as follows.

1) Area—$A(f)$: The area cost function was proposed by Kerkhoff and Robroek [7] and Robroek [11] as a way to minimize implementation costs of CCD circuits, especially chip area. The cost $A(f)$ of a specific function f is determined by the best realization known at that time. As improved realizations become available, this function changes. In this paper, we use the costs derived originally in [7] and [11] with improvements listed in [8] and [13].

2) Transition Count—$TC(f)$: The transition count was proposed in [8] as a simpler alternative to the area cost. Unlike area cost, the transition count is not derived from a table but is computed directly from the function. However, as noted in [8], there is a correlation between the area cost of a function and its transition count. The transition count, $TC(f)$, of a function f is the number of times the logic value in f changes from decreasing to increasing and vice versa plus 1 if the function is initially decreasing, as the input logic values x increase from 0 to 3. For example, $TC(\langle 1,1,2,2\rangle) = 0$ and $TC(\langle 2,0,3,1\rangle) = 3$.

Formally, given $f(x) = \langle a_0, a_1, a_2, a_3\rangle$, let

$$I_i(f) = \begin{cases} 1 & \text{if } a_{i-1} < a_i > a_{i+1} \text{ or } a_{i-1} > a_i < a_{i+1}, \\ & \qquad 1 \leq i \leq 2 \\ 0 & \text{otherwise,} \end{cases}$$

$$I_{12}(f) = \begin{cases} 1 & \text{if } a_0 < a_1 = a_2 > a_3 \text{ or } a_0 > a_1 = a_2 < a_3 \\ 0 & \text{otherwise, and} \end{cases}$$

$$ID(f) = \begin{cases} 1 & \text{if there is a } p,\ 0 \leq p \leq 2, \text{ such that} \\ & \quad a_0 = a_1 = \cdots = a_p > a_{p+1}, \\ 0 & \text{otherwise.} \end{cases}$$

$I_i(f)$ is 1 iff the ith function value is either strictly larger or strictly smaller than both of the two adjacent values, for $i = 1$ or 2. That is, I_i is 1 if there is an inflection point. As such, values on both sides of a prospective inflection point must be known. Thus, i is restricted to interior logic values. $I_{12}(f)$ is 1 iff the middle two logic values are the same and are either strictly larger or strictly smaller than both of the end values. $ID(f)$ is 1 iff the function values are initially decreasing.

The *transition count* $TC(f)$ of function f is

$$TC(f) = I_1(f) + I_2(f) + I_{12}(f) + ID(f).$$

3) Total Transition Size—$TTS(f)$: The correlation between transition count and area cost is not exact. To achieve a closer correlation, the total transition size was introduced [13]. In the transition count, for each transition from increasing to decreasing or from decreasing to increasing, 1 is added to the function's cost, whereas, with total transition size, the exact size of the transition is added to the cost. That is, $TTS(f)$ is the sum of the size of each transition (increasing to decreasing or decreasing to increasing) plus the size of the first transition (again) if f is initially decreasing. For example, $TTS(\langle 1,1,2,2\rangle) = 1$ and $TTS(\langle 2,0,3,1\rangle) = 9$.

Formally, given $f(x) = \langle a_0, a_1, a_2, a_3\rangle$, define beginning, middle, and end transition sizes as follows,

$$S_b(f) = \begin{cases} |a_1 - a_0| & \text{if } I_1 = 1 \text{ or } I_{12} = 1 \\ |a_2 - a_0| & \text{if } I_1 = 0 \text{ and } I_2 = 1 \\ |a_3 - a_0| & \text{otherwise,} \end{cases}$$

$$S_m(f) = \begin{cases} |a_2 - a_1| & \text{if } I_1 = I_2 = 1 \\ 0 & \text{otherwise, and} \end{cases}$$

$$S_e(f) = \begin{cases} |a_3 - a_2| & \text{if } I_2 = 1 \text{ or } I_{12} = 1 \\ |a_3 - a_1| & \text{if } I_2 = 0 \text{ and } I_1 = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$S_b(f)$, $S_m(f)$, and $S_e(f)$ are the beginning, middle, and end transition sizes, respectively. The *total transition size* $TTS(f)$ of a function f is

$$TTS(f) = S_b(f) + S_m(f) + S_e(f) + S_b(f)ID(f).$$

Since total transition size can be measured between each adjacent pair of logic values, an alternative definition is

$$TTS(f) = |a_1 - a_0| + |a_2 - a_1| + |a_3 - a_2| + S_b(f)ID(f).$$

4) Constant—$C(f)$: For this cost function, each function f has a fixed cost, $C(f) = c$. We consider $c = 0$, in which case, the cost of realizing a given function is just the cost of combining cost-table functions. Such a cost function approximates the situation where the cost of combining cost-table functions is much larger than the cost of the functions themselves.

5) Sum—$SC(f)$: The surprisingly similar behavior of the above four cost functions, with respect to the dependence of costs of near-minimal cost-tables on cost-table size, inspired an examination of a significantly different cost function. With sum cost, the cost of a function $f(x)$ is the arithmetic sum of the logic values produced when $x = 0, 1, 2, 3$. For example, $SC(\langle 1, 1, 2, 2\rangle) = 6$ and $SC(\langle 2, 0, 3, 1\rangle) = 6$.

Formally, given $f(x) = \langle a_0, a_1, a_2, a_3\rangle$ the *sum cost* is

$$SC(\langle a_0, a_1, a_2, a_3\rangle) = a_0 + a_1 + a_2 + a_3.$$

III. The Minimal Cost-Table Problem

In preparation for the discussion in the next section on the results of heuristics for designing minimal cost-tables, we show here the context in which such heuristics operate. That is, a heuristic can be viewed as simply a selection of a cost-table from all possible cost-tables. We show the range over which the selection is made. In so doing, we determine the importance of finding good heuristics. A criteria by which a cost-table is judged is the average function cost over all functions realized. Equivalently, the total cost over all functions can be used, which we do here. Schueller, Tirumalai, and Butler [13] plot the total cost of both near-minimal and near-maximal cost-tables as a function of cost-table size for each of the five cost functions defined in the previous section. It is shown that the total cost of near-minimal cost-tables decreases rapidly as the table size t increases, when t is small. However, when t is large, the decrease is small, and there is little (and sometimes no) benefit to increasing the size of the cost-table. On the other hand, the total cost of near-maximal cost-tables is shown to decrease almost linearly as the size of the cost-table increases. Thus, there is a large difference between near-minimal and near-maximal cost-tables when size is small and a small difference for large cost-tables. To analyze the merits of heuristics, we seek the distribution of the total costs of cost-tables over all cost-tables.

We know of no computationally feasible method for finding this distribution. Exhaustive enumeration is infeasible even for one-variable 4-valued functions. For example, for size $t = 129$, there are $\binom{251}{t-5} \approx 10^{74}$ cost-tables. Therefore, our approach is to randomly generate sample cost-tables of a specific size and then find the distribution of the total costs of these samples. The results are shown in Fig. 1. In computing these costs, we assume that the cost, s, of the two-input adder is 2 for all cost functions. The total cost for near-minimal and near-maximal cost-tables found in [13] are plotted in the horizontal plane. Specifically, the axis pointing into the page (northeast) represents total cost T, which is a function of t, the cost-table size, and is represented by the axis pointing down and to the right (southeast). So, a point on the near-minimal curve represents the total cost of the cost-table with the smallest known cost, while a point on the near-maximal curve represents the total cost of a cost-table with the largest known cost. It can be seen that, for all five cost functions, as cost-table size increases, the total cost of the near-minimal cost-table drops sharply until about size 20, after which there is only a marginal decrease in total cost as size increases. The heavier lines associated with the near-minimal costs represent *known* minimal cost-tables. From this, it can be seen that the majority of the near-minimal cost-tables are known to be minimal. Also plotted are the average cost and the average ± one standard deviation for a sample set of 500 cost-tables. That is, for each cost-table size, 500 randomly selected cost-tables are generated, and the average and standard deviation computed. The axis pointing up (north) shows the number N of occurrences of cost-tables at the various sizes. To avoid an overly complex diagram, only seven distributions are shown. These are for cost-table sizes 32, 64, 96, 128, 160, 192, and 224. Each vertical line represents the number of cost-tables whose total cost occurs in an interval called a *cell*. Because of the wide variation in costs among the cost functions, the cell sizes are normalized to one one-hundredth of the cost of the basis cost-table. Without normalization, cost functions with high costs and thus many different costs, such as area, produce histograms with many short, indistinguishable lines (this corresponds to a cell size of 1).

While 500 is a relatively small sample set size (there can be as many as 10^{74} cost-tables for each of the chosen sizes), the complexity of calculating the total cost of individual cost-tables limits the number of samples we can generate. However, we have run our programs with smaller sample sizes, and the results are similar, suggesting that our sample size is sufficiently large. Each calculation of total cost requires a nearly exhaustive search.

The plots for all five cost functions have similar features. One of the most interesting is the small standard deviation over all sizes; almost all cost-tables are near average. The small standard deviation is especially surprising for small cost-tables, where there is a large difference in cost between near-minimal and near-maximal cost-tables.

For large cost-tables, there is little difference in cost between an average cost-table and a near-minimal cost-table for all five cost functions. This shows that most heuristics work well for large cost-tables. On the other hand, a significant difference exists between the average cost and the near-minimal tables for small size, which implies heuristics for generating small cost-tables should be chosen carefully. This observation is important, since practical cost-tables on more than one variable are small compared to the set of all functions. For example, since there are 4×10^9 two-variable 4-valued functions, any cost-table small enough to be stored in a modern computer is small relative to the number of all possible functions. This observation is the basis for the statement earlier that it is important to have good heuristics for designing cost-tables.

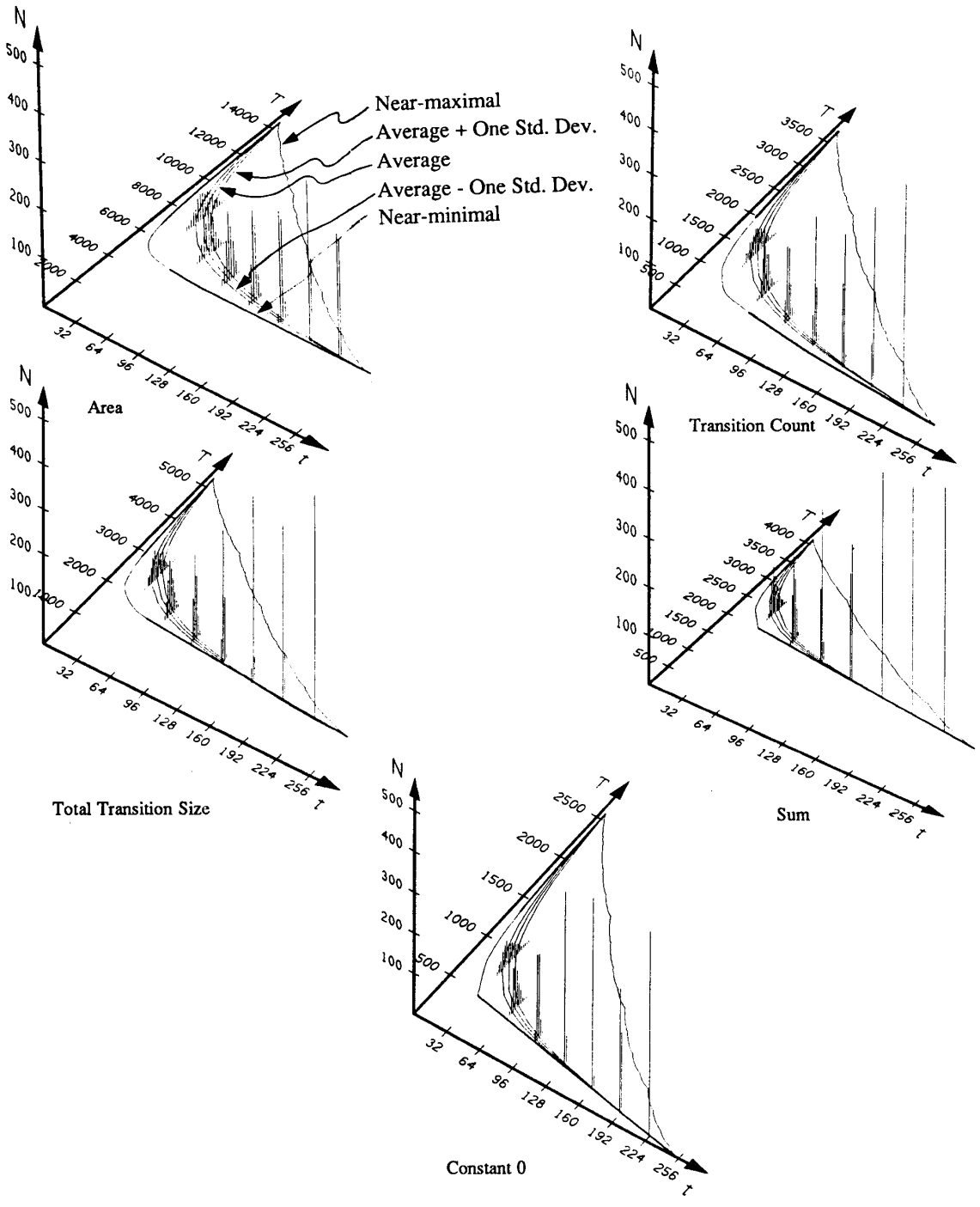


Fig. 1. Distribution of random cost-tables for one-variable 4-valued functions with respect to total cost and size. Each vertical line represents the number N of cost-tables with total cost T and size t. The three solid lines under the distributions represent the average total cost and the average $\pm$ one standard deviation.

IV. Analysis of Heuristics for Finding Minimal Cost-Tables

Four heuristic algorithms for finding cost-tables are analyzed. They are:

1) MAXIMUM_REDUCTION: Consider BT, the basis cost-table and some given cost function c. Given a function f that has a realization of less cost than a realization from BT (i.e.,

$c(f) < c_{BT}(f)$), the addition of f to BT yields a lower total cost. That is, any use of f achieves a reduction in cost that is the difference between the cost of f in BT and the cost of f in $BT \cup \{f\}$. Let the *reduction* of f be

$$\gamma(f) = c_{BT}(f) - c_{BT\cup\{f\}}(f).$$

The MAXIMUM_REDUCTION heuristic forms a cost-table of size t by combining BT with the $t - |BT|$ functions of largest $\gamma(f)$, with ties broken arbitrarily.

2) MAXIMUM_USE: Let $\Psi(f)$ be the total number of times f can be used in the realization of functions in $U_{n,r}$. That is,

$$\Psi(f) = \sum_{g\in U_{n,r}} \rho(f,g)$$

where $\rho(f,g)$ is the number of times f can be used in the realization of g. Specifically, $\rho(f,g)$ is the largest integer such that $g(\boldsymbol{v}) \geq \rho(f,g)f(\boldsymbol{v})$ for all assignments $\boldsymbol{v}$ of values to the variables. For example, if $f(x) = \langle 1,1,1,1\rangle$ and $g(x) = \langle 3,2,3,2\rangle$, then $\rho(f,g) = 2$. That is, $f(x)$ can be used at most two times in $g(x)$ ($\langle 3,2,3,2\rangle = \langle 1,1,1,1\rangle + \langle 1,1,1,1\rangle + \langle 1,0,1,0\rangle$). The MAXIMUM_USE heuristic forms a cost-table of size t by combining BT with the $t - |BT|$ functions of largest $\Psi(f)$, with ties broken arbitrarily.

3) MAXIMUM_TOTAL_REDUCTION: Given f, the *total reduction* achieved by using f is

$$R_{BT}(f) = \gamma(f)\Psi(f).$$

That is, adding f to the basis cost-table BT produces a total cost that is less than $T(BT)$ by $R_{BT}(f)$. Thus, a minimal cost-table of size $|BT| + 1$ is achieved with $BT \cup \{f\}$, where f is a function of largest $R_{BT}(f)$. The MAXIMUM_TOTAL_REDUCTION heuristic forms a cost-table of size t by combining BT with the $t - |BT|$ functions of largest $R_{BT}(f)$, with ties broken arbitrarily. While this heuristic produces the minimal cost-table of size $|BT| + 1$, it is not guaranteed to produce minimal cost-tables of larger size. This is discussed later.

4) ITERATIVE_BEST: A near-minimal cost-table of size $t + 1$ can be formed from a near-minimal cost-table F of size t by choosing a function not in F and adding it to F. If we compute the total cost of all cost-tables so formed and keep the one with lowest cost, we are likely to achieve a total cost that is close to minimal. However, we can improve the chances of finding a minimal cost-table by performing this process on two or more near-minimal cost-tables, instead of just one. This is what is done in calculating the near-minimal cost curves for Fig. 1. Here, the $d = 10$ best cost-tables were retained. The formal algorithm is given in [13]. Making d as large as possible improves the results of this heuristic. We observe that the marginal improvement drops off rapidly as d increases from 1.

Fig. 2 shows how the four heuristics compare. Shown are the worst costs from [13], the costs produced by the four heuristics, and the average costs produced by the statistical study. The MAXIMUM_REDUCTION heuristic produces poor results, significantly worse than even the average for randomly chosen cost-table. The heuristic consistently producing the lowest total cost is ITERATIVE_BEST, with MAXIMUM_USE and MAXIMUM_TOTAL_REDUCTION doing considerably better than the average case. For small and mid-size cost-tables, the latter curves fluctuate because of the random choice of cost-tables when ties are broken. For the area cost, it is interesting that the cost-tables of Kerkhoff and Robroek [7], [11] and Lee and Butler [8], both chosen heuristically, are better than any of the random cost-tables generated, but are about the same as the costs produced by the MAXIMUM_USE and MAXIMUM_TOTAL_REDUCTION heuristics.

In understanding these results, we note particularly the poor performance of MAXIMAL_REDUCTION. In this heuristic, a function f with a large reduction, $\gamma(f) = c_{BT}(f) - c_{BT\cup\{f\}}(f)$, is included before functions with smaller reduction values. In the case of all five cost functions, $\langle 3,3,3,3\rangle$ has the largest reduction. Thus, in MAXIMUM_REDUCTION, it is included in *all* cost-tables of size 6 (1 larger than BT). However, it is an unfortunate choice, since there is only one realization where it is used, specifically $\langle 3,3,3,3\rangle$. Thus, there is only a marginal improvement in the total cost over the basis cost-table. To see this, consider, for example, the transition count. For $f(x) = \langle 3,3,3,3\rangle$, $TC(f) = 0$, and thus $c_{BT\cup\{f\}}(f) = 0$. Further, $c_{BT}(f) = 3(1 + 1 + 1 + 0) + (12 - 1)2 = 31$, and so $\gamma(f) = c_{BT}(f) - c_{BT\cup\{f\}}(f) = 31$. This is the largest reduction of any function g, since there is no larger $c_{BT}(g)$, nor is there a smaller $c_{BT\cup\{g\}}(g)$. Thus, MAXIMUM_REDUCTION yields $BT \cup \{\langle 3,3,3,3\rangle\}$ as the cost-table of size 6. The total cost for the basis cost-table using the transition count is $T(BT) = 3714$, while $T(BT \cup \{\langle 3,3,3,3\rangle\}) = 3683$. Thus, adding $\langle 3,3,3,3\rangle$ to the basis cost-table nets only a 0.8% improvement. A provably minimal cost-table of size 6 consists of BT and $\langle 1,1,1,1\rangle$ and has a total cost of $T(BT \cup \{\langle 1,1,1,1\rangle\}) = 2832$, resulting in a 23.7% improvement. Interestingly, $BT \cup \{\langle 3,3,3,3\rangle\}$ is a provably *maximal* cost-table.

While we have performed this analysis for cost-tables of size 6, clearly a similar trend exists for larger cost-tables. A good function to add to an existing cost-table is one that 1) has low cost and 2) can be used in the realization of many other functions. With MAXIMUM_REDUCTION, the chosen functions tend to satisfy the first criteria but not the second.

Heuristic MAXIMUM_USE, which does significantly better than MAXIMUM_REDUCTION, satisfies the second criteria. That is, a function is added the current cost-table if it can be used in the realization of the most number of functions, $\Psi(f) = \sum_{g\in U_{n,r}} \rho(f,g)$. For example, except for the basis functions, the functions usable in the most other functions are $\langle 1,1,0,0\rangle$, $\langle 1,0,1,0\rangle$, $\langle 1,0,0,1\rangle$, $\langle 0,1,1,0\rangle$, $\langle 0,1,0,1\rangle$, and $\langle 0,0,1,1\rangle$. Thus, with the MAXIMUM_USE heuristic, the cost-table of size 6 contains the basis functions and one of the six functions with two 1's. The specific function is chosen randomly. Because MAXIMUM_USE depends only on the relationship among functions, the cost-tables generated are the same for all cost functions.

While a function with two 1's is a reasonably good choice for the smallest nonbasis cost-table, it is not the best choice. For all five cost functions, a provably

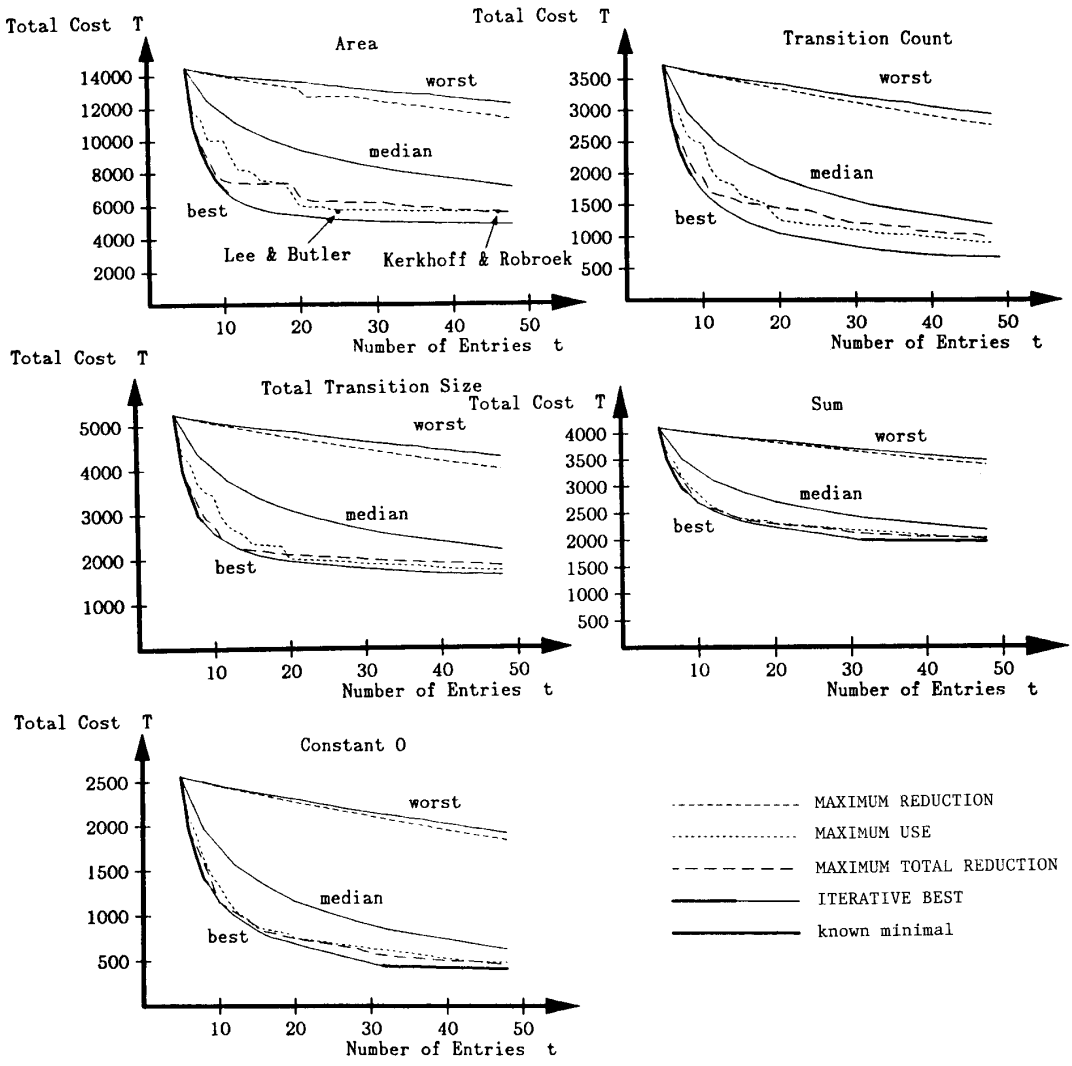


Fig. 2. The results of four heuristic algorithms for finding minimal cost-tables on one-variable 4-valued functions. The three solid lines represent the worst, median, and ITERATIVE_BEST plots, while the three dotted lines represent the MAXIMUM_REDUCTION, MAXIMUM_USE, and MAXIMUM_TOTAL_REDUCTION plots.

minimal cost-table of size 6 is $BT \cup \{\langle 1,1,1,1\rangle\}$. MAXIMUM_TOTAL_REDUCTION, however, correctly chooses $\langle 1,1,1,1\rangle$ as the function to be added to the basis cost-table. In this heuristic, the function chosen is the one with maximum $R_{BT}(f) = \gamma(f)\Psi(f)$. Indeed, $T(BT \cup \{f\}) = T(BT) - R_{BT}(f)$, and so, the smallest $T(BT \cup \{f\})$ is achieved for a function f with the largest $R_{BT}(f)$. As was observed earlier, while the use of MAXIMUM_TOTAL_REDUCTION generates the minimal cost-table of size $|BT| + 1$, it does not necessarily generate minimal cost-tables of larger size. This is because it neglects the interaction among cost-table functions. As with all previous heuristics, once a function is chosen for a cost-table of size t, it is included in *all* larger cost-tables. This can result in nonminimal cost-tables. For example, while $\langle 1,1,1,1\rangle$ is the best function to use in a cost-table of size 6, for the transition count, total transition size, constant 0, and sum cost functions, the (only) minimal cost-table of size 7 does *not* contain $\langle 1,1,1,1\rangle$; instead $\langle 1,1,0,0\rangle$ and $\langle 1,0,1,1\rangle$ are included.

ITERATIVE_BEST produces the lowest cost because it accommodates the interaction among functions. In this heuristic, the d cost-tables of size t with lowest total cost are used to generate d cost-tables of size $t + 1$ with lowest cost. Specifically, for each of the d cost-tables of size t with lowest cost, one remaining function is added forming a size $t + 1$ cost-table. Among all cost-tables of size $t + 1$ so formed, the d least costly are chosen.

In our application of this heuristic, $d = 10$. Over the whole range of t, a total of 316 260 cost-tables are examined. This is considerably less than the 2^{251} cost-tables considered in exhaustive enumeration. If two of the t-entry cost-tables

TABLE I
RANGE OF COST-TABLE SIZES WHERE ITERATIVE_BEST GENERATES A PROVABLY MINIMAL COST-TABLE

Cost Function	Cost-Table Size For Which ITERATIVE_BEST Produces a Provably Minimal Cost-Table	Cost-Table Size For Which ITERATIVE_BEST Produces a Nonminimal Cost-Table
Area	5–10 and 52–256	–
Transition Count	5–7 and 67–256	–
Total Transition Size	5–7 and 52–256	–
Constant 0	5–7 and 32–256	31
Sum	5–7 and 32–256	31

among those d with lowest total cost are the same except for one function, then a $t+1$-entry cost-table is generated twice. However, an upper bound on the amount of double counting is a small fraction of the total, and, although our program generates these, only a small penalty in computation time is paid.

The superiority of ITERATIVE_BEST is shown by the number of cost-tables it generates that are provably minimal. Table I (from [13]) shows the range of cost-table size where provably minimal cost-tables are generated. The heavy lines in Figs. 1 and 2 associated with ITERATIVE_BEST correspond to ranges where provably minimal cost-tables are generated. The range of lower cost-table sizes of minimal cost-tables has been shown by exhaustive enumeration. Specifically, all potentially minimal cost-tables of these sizes have been generated and checked. For larger sizes, exhaustive enumeration is too time consuming. The range of larger cost-table sizes in Table I corresponding to minimal cost-tables has been proved so in Lemma 1 of [13]. Specifically, this is a sufficient condition for a cost-table to be minimal. The table also shows that for one cost-table size, a *nonminimal* cost-table is produced. For all values not shown, it is not known whether the generated cost-tables are minimal.

V. THE LINEAR COST

As observed in [13], the plots for the sum cost and the constant 0 cost are similar. Further, it was observed that if F is a near-minimal or near-maximal cost-table using constant 0 cost, then F is also a near-minimal or near-maximal cost-table using sum cost, and vice-versa. We now generalize this observation to n-variable, r-valued functions and relax a condition on the way such cost functions are formulated.

Let $f \in U_{n,r}$ be an n-variable, r-valued function, and let $q = r^n$. Then, f can be represented as a vector with q components: $f = \langle a_0, a_1, \cdots, a_{q-1} \rangle$. Let F be a cost-table, and define a *linear* cost of a function f as follows:

$$LC(f) \equiv \sum_{i=0}^{q-1} c_i a_i + c_q$$

where c_i is a real-valued constant. For example, with $q = r^n = 4^1$, $c_0 = c_1 = c_2 = c_3 = 1$ and $c_4 = 0$ corresponds to the sum cost function, and $c_0 = c_1 = c_2 = c_3 = c_4 = 0$ corresponds to the constant 0 cost function.

Let $L_F(f)$ be the cost of a minimal realization for function f with respect to cost-table F, and let a minimal realization be $f = f_1 + f_2 + \cdots + f_m$, where $f_i \in F$. Then,

$$L_F(f) = \sum_{j=1}^{m} LC(f_j) + (m-1)s,$$
$$= \sum_{j=1}^{m} \left[\sum_{i=0}^{q-1} f_{j_i} c_i + c_q \right] + (m-1)s$$

where f_{j_i} is the value of f_j for the ith assignment of values to the variable. Rearranging this expression yields

$$L_F(f) = \sum_{i=0}^{q-1} \sum_{j=1}^{m} [f_{j_i} c_i] + m c_q + (m-1)s,$$
$$= \sum_{i=0}^{q-1} a_i c_i + m c_q + (m-1)s, \quad \text{since } \sum_{j=1}^{m} f_{j_i} = a_i,$$
$$= LC(f) + (m-1)(c_q + s).$$

Let $ADD_F(f)$ be the number of adders used in a minimal realization of function f with respect to cost-table F. Then,

$$L_F(f) = LC(f) + ADD_F(f)(c_q + s).$$

The minimal cost realization for f is $LC(f)$, the cost of f in the cost-table containing all functions, plus $(c_q + s)$ times the number of adders in a minimal realization. If $(c_q + s) > 0$, then minimizing the cost of f is the same as minimizing the number of adders used in its realization, *regardless of the values of the constants* c_i, for $0 \le i < q$. Thus, the minimal realization of f does *not* depend on the linear cost function used. This topic is discussed in more detail in Butler and Schueller [5]. For example, it is shown that if the cost of adding functions is sufficiently large, *any* cost function yields a minimal realization that is identical to a minimal realization under a linear cost function.

The total cost of a cost-table F is

$$T(F) = \sum_{f \in U_{n,r}} L_F(f),$$
$$= \sum_{f \in U_{n,r}} [LC(f) + ADD_F(f)(c_q + s)],$$
$$= T(U_{n,r}) + (c_q + s) \sum_{f \in U_{n,r}} ADD_F(f).$$

Since $T(U_{n,r})$, c_q, and s are constants, minimizing the total cost is equivalent to minimizing the total number of adders needed to realize *all* functions, if $c_q + s > 0$. Therefore, if $c_q + s > 0$, a minimal cost-table of size t corresponds to a set of t functions that sum to form *all* functions with the least number of adders (independent of the linear cost parameters!). This proves the following.

Lemma 1: Let LC and LC' be two linear cost functions with constant components c_q and c'_q, respectively, such that $c_q + s > 0$ and $c'_q + s > 0$, where s is the cost of adding two cost-table functions. Cost-table F is a minimal cost-table using LC iff F is a minimal cost-table using LC'.

The sum and constant 0 costs considered earlier are instances of this linear cost. For both the sum cost and the

constant 0 cost function, $c_q = 0$. Since the cost of the adder s is a constant 2, $c_q + s = 2$, and $(c_q + s) \sum_{f \in U_{n,r}} ADD_F(f)$ is the same for both cost functions. Thus, the only difference in total costs between these linear cost functions is the difference in the costs of the universal cost-table. For the constant 0 cost function, the total cost is 0. For the sum cost, the total cost $T_{SC}(U_{1,4})$ is

$$\begin{aligned} T_{SC}(U_{1,4}) &= \sum_{f \in U_{1,4}} SC(f) = \sum_{f \in U_{1,4}} \sum_{i=0}^{3} a_i \\ &= \sum_{i_0, i_1, i_2, i_3 \in \{0,1,2,3\}} i_0 + i_1 + i_2 + i_3 \\ &= 1536 \end{aligned}$$

which is the difference in costs between the sum cost and the constant 0 cost shown in the plots of Fig. 1 for these two cost functions.

We find the independence of the composition of minimal cost-tables on the linear cost parameters c_i to be surprising. Intuitively, the best functions for inclusion in a cost-table for any cost function are those that have 1) low cost and 2) can be used many times in the realization of other functions. This is confirmed by the list of the minimal cost-tables in [13], which shows that the best functions for the cost functions considered tend to consist of 0's and 1's exclusively (for all the cost functions considered, functions with 0's and 1's have low cost). It is interesting to note that, since the constants c_i for $0 \leq i \leq q-1$ in the linear cost function can be negative, a cost function exists where functions with 0's and 1's are the most expensive functions. However, from Lemma 1, the minimal cost-table is unaffected by specific values of c_i. Therefore, these (expensive) functions still are the best functions to use. Thus, of the two criteria for selecting the best functions to include, the number of times a function can be used is more important. This is the second substantiation of this observation. Recall that, of the two heuristics for generating cost-tables, MAXIMUM_USE, which selects a function that is potentially usable in the most number of other functions, is significantly better than MAXIMUM_REDUCTION, which selects functions with the largest reduction, a parameter that is directly related to function cost.

VI. The Presence of Composite Functions in Minimal Cost-Tables

In the design of a cost-table, it is useful to know if certain functions never occur in minimal cost-tables. For example, the design of binary sum-of-products expressions for realization by programmable logic arrays relies on a significant reduction in the search by the observation that only prime implicants need be considered. It is tempting to believe that certain functions that are better realized as the composition of other functions never occur in a minimal cost-table. We show, however, that this is not so. For a given cost function c, a cost-table F, and a function f, f is said to belong to one of three *composition classes*. f is

1) *noncomposite with respect to* F iff $c(f) < c_F(f)$,
2) *simple composite with respect to* F iff $c(f) = c_F(f)$, and
3) *pure composite with respect to* F iff $c(f) > c_F(f)$.

A function that is simple or pure composite with respect to F is said to be *composite* with respect to F. If f is noncomposite with respect to F, then the cost of f is reduced by adding f to cost-table F. This, in turn, may reduce the cost of other functions that can use f in their realizations. Adding a function f that is simple composite with respect to F to the cost-table does not alter the cost of any function; it simply enlarges the cost-table. Similarly, if f is pure composite with respect to F, there is also no benefit to adding f to cost-table F. By definition, a lower cost realization exists, so $c(f)$ is not the cost of a minimal realization. In spite of this, such functions do occur as the result of mathematical formulations of cost functions. For example, the compensated transition count (CTC) cost of Tirumalai [16] is such a cost. The CTC of a function is the sum of the (second) transition sizes when the function changes from increasing to decreasing, or vice-versa, plus the size of the first transition, if the function is initially decreasing. Consider one-variable 4-valued functions $f(x) = \langle 1,1,0,3 \rangle$, $f_1(x) = \langle 1,1,0,0 \rangle$, and $f_2(x) = \langle 1,0,0,3 \rangle$. $CTC(f) = 4$, $CTC(f_1) = 1$, and $CTC(f_2) = 0$. The cost of the realization $f_1(x) + f_2(x)$ is $CTC(f_1) + CTC(f_2) + s = 1 + s$. If $f_1, f_2 \in F$ and $s < 3$, then f is a pure composite function with respect to F. Using the CTC cost and an adder cost of 2, there are 36 pure composite and 44 simple composite functions with respect to U_{14}. In general, the number of composite functions depends on the cost of combining functions.

In our previous discussion of the linear cost function, we assumed that $(c_q + s) > 0$. We now consider the effect of relaxing this condition. Recall that $L_F(f)$, the cost of a function f using cost-table F, is

$$L_F(f) = LC(f) + ADD_F(f)(c_q + s).$$

Lemma 2: Let L be a linear cost function, where c_q is the constant parameter. Let F be a cost-table used with LC, and let s be the cost of adding two functions. A function $f \notin F$ is

1) noncomposite with respect to F iff $(c_q + s) > 0$,
2) simple composite with respect to F iff $(c_q + s) = 0$, and
3) pure composite with respect to F iff $(c_q + s) < 0$.

Proof: Since $f \notin F$, any realization of f with respect to cost-table F must be a composition of functions, so $ADD_F(f) > 0$. The proof follows directly from the definitions of composite classes. Q.E.D.

Applying Lemma 2 with $F = BT$, we find that all functions $f \in U_{n,r} - BT$ belong to the same composition class, regardless of the cost-table. Here, $(c_q + s) = 0$ and all possible realizations of a function have the same cost. As a result, all cost-tables have the same total cost, regardless of size. Under this condition, any attempt to minimize the cost of a function is futile. If $(c_q + s) < 0$, there is a paradox; the cost of *any* realization of a function can be reduced simply by adding the function consisting of all 0's!

A function is *noncomposite*, *simple composite*, *pure composite*, or *composite*, if it is noncomposite, simple composite,

pure composite, or composite, respectively, with respect to the universal cost-table. The search for minimal cost-tables would be faster if cost-table functions were all noncomposite. However, this is not the case. For example, the universal cost-table is a minimal cost-table that can contain a composite function. In the universal cost-tables for the area and transition count cost functions, there are 171 and 6 simple composite functions, respectively. However, in the universal cost-table with total transition size, sum, or constant 0 function, there are no simple composite functions. There are no pure composite functions in any universal cost-tables for the five cost functions considered.

Composite functions provide no benefit to the universal cost-table. Thus, $T(U_{n,r} - \{f\}) = T(U_{n,r})$, where f is composite, and it follows that $U_{n,r} - \{f\}$ is minimal also. Further, $T(U_{n,r} - \{f | f \text{ is composite}\}) = T(U_{n,r})$, and so $U_{n,r} - \{f | f \text{ is composite}\}$ is a minimal cost-table with no composite functions. One might at first believe that minimal cost-tables of size $t < |U_{n,r} - \{f | f \text{ is composite}\}|$ also contain no composite functions. However, this is not the case.

Observation: There exists a minimal cost-table of size 1 larger than the basis cost-table that contains a composite function.

Proof: Consider the set of one-variable six-valued functions, $U_{1,6}$. Let $c(b_i) = 10$, where b_i is the basis function whose ith component is 1. Let the cost of the function $\langle 0,0,0,0,0,0 \rangle$ be 0, and let $s = 2$. Let there be three noncomposite functions with respect to BT, $f_1(x) = \langle 2,0,0,0,0,0 \rangle$, $f_2(x) = \langle 0,2,0,0,0,0 \rangle$, and $f_3(x) = \langle 2,2,0,0,0,0 \rangle$, with $c(f_1) = c(f_2) = 15$ and $c(f_3) = 33$. Assume $c(f) \geq c_{BT}(f)$ for $f \in U_{1,6} - BT - \{f_1, f_2, f_3\}$. f_3 can be realized as $f_1 + f_2$ at a cost of $15 + 15 + 2 = 32$, which is less than $c(f_3) = 33$. Thus, f_3 is pure composite with respect to any cost-table containing $\{f_1, f_2\}$. There are only three possibilities for minimal cost-tables of size $|BT| + 1$. Since $T(BT \cup \{f_1\}) = 7978178$, $T(BT \cup \{f_2\}) = 7978178$, and $T(BT \cup \{f_3\}) = 7967810$, the best cost-table of size 8 is $BT \cup \{f_3\}$, which contains a function that is pure composite. Q.E.D.

VII. Minimal Cost Tables of Size $|BT| + 1$

It is surprisingly difficult to find *provably* minimal cost-tables. The problem of finding a minimal cost realization of a given function by cost-table is known to be NP-complete [14], and it is likely that the problem of finding a minimal cost-table is also NP-complete. Even for small sizes, considerable computation is needed to find provably minimal cost-tables. The negative result on composite functions shows that not even these functions can be removed from consideration. For a special case, however, we are able to describe, in a precise way, the minimal cost-table. In this section, we consider, for the linear cost, minimal cost-tables that have size one larger than that of the basis cost-table. Rather than calculating the total cost directly, we proceed by considering the reduction $R_{BT}(f)$ in the total cost of cost-table formed by adding function f to BT. We have $R_F(f) = T(F) - T(F \cup \{f\})$. However, we are interested in a form of $R_F(f)$ more like that of the previous section. Assuming the function consisting of all 0's is used only in its own realization, the realization of any function f using just basis functions is unique. Let a_i be the ith component of f and b_i is the basis function whose ith component is 1.

$$\begin{aligned} c_{BT}(f) &= \sum_{i=0}^{q-1} [c(b_i)a_i] + sADD_{BT}(f), \\ &= \sum_{i=0}^{q-1} [c(b_i)a_i] + s\left[\sum_{i=0}^{q-1} [a_i] - 1\right] \end{aligned}$$

where the second expression follows from the first by the observation that the number of two-input adders needed to realize f is the sum of the components in f less 1.

$R_{BT}(f)$ is the product of the reduction in cost each time f is used and the number of times f is used in the realization of functions. The reduction resulting from one use of f is given as

$$\begin{aligned} \gamma(f) &= c_{BT}(f) - c_{BT \cup \{f\}}(f), \\ &= \begin{cases} c_{BT}(f) - c(f) & \text{if } f \text{ is noncomposite with} \\ & \quad \text{respect to } BT \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Let $\Psi(f)$ be the total number of times f is used in the realization of functions in $U_{n,r}$, and let $\Phi_{\geq l}(f)$ be the number of functions that can use function f at least l times in their realizations; that is,

$$\Phi_{\geq l}(f) = \prod_{i=0}^{q-1} (r - la_i).$$

Since addition is undefined if a component sum exceeds $r-1$, there is a limit to the number of times f can be used in the realization of functions. Let κ be the maximum component of f, i.e., $\kappa = \max_{0 \leq j \leq q-1}\{a_j\}$, and $0 \leq \kappa \leq r-1$. Function f can be used at most $\lfloor \frac{r-1}{\kappa} \rfloor$ times in the realization of any function. Consequently, $\Phi_{\geq l}(f) = 0$, for $l > \lfloor \frac{r-1}{\kappa} \rfloor$.

$$\begin{aligned} \Psi(f) &= \sum_{l=1}^{\lfloor \frac{r-1}{\kappa} \rfloor} l[\Phi_{\geq l}(f) - \Phi_{\geq l+1}(f)], \\ &= \sum_{l=1}^{\lfloor \frac{r-1}{\kappa} \rfloor} \Phi_l(f), \\ &= \sum_{l=1}^{\lfloor \frac{r-1}{\kappa} \rfloor} \prod_{i=0}^{q-1} (r - la_i). \end{aligned}$$

$\Psi(f)$ is independent of the cost function used, while $\gamma(f)$ does depend on the cost function. Using the linear cost with $(c_q + s) = 1$ with c_i unrestricted,

$$\begin{aligned} \gamma(f) &= L_{BT}(f) - LC(f), \\ &= ADD_{BT}(f), \\ &= \sum_{i=0}^{q-1} (a_i) - 1. \end{aligned}$$

Therefore,

$$R_{BT}(f) = \gamma(f)\Psi(f),$$
$$= \left[\sum_{i=0}^{q-1}(a_i) - 1\right] \sum_{l=1}^{\lfloor \frac{r-1}{\kappa} \rfloor} \prod_{j=0}^{q-1}(r - la_j).$$

We now determine the best function to add to BT.

Theorem 1: Using the linear cost, a minimal cost-table of size $|BT| + 1$ is $BT \cup \{f\}$, where $f : R^n \rightarrow \{0, 1\}$.

Proof: On the contrary, assume f has at least one component f_i, such that $f_i > 1$. We show there is another function g such that $R_{BT}(g) > R_{BT}(f)$, where $g_i = 1$. It follows that $BT \cup \{g\}$ is a cost-table of lower cost than $BT \cup \{f\}$, a contradiction.

g is derived from f by a sequence of operations, first of Type 1 and then, if necessary, of Type 2. Let f' be f or some intermediate function. The *Type 1* operation is applied to pairs of logic values in f', one 0 and the other > 1, with the former increased by 1 and the latter decreased by 1. If successive applications yields a function with exclusively 0's and 1's, we are done. Otherwise, the final function consists of no 0's and at least one nonzero value greater than 1. The *Type 2* operation reduces by 1 each nonzero logic value greater than 1. A succession of these operations yields the constant function $g = 1$. The theorem follows from a demonstration that Type 1 and 2 operations both yield a function with a larger reduction.

Consider the Type 1 operation. The reduction of an intermediate function f' can be expressed as

$$R_{BT}(f') = \gamma(f') \sum_{l=1}^{\lfloor \frac{r-1}{\kappa} \rfloor} \left[\prod_{\substack{k=0 \\ k \neq i,j}}^{q-1} (r - lf_k')\right] (r - lf_i')(r - lf_j')$$

where $f_i' = 0$ and $f_j' > 1$. On the application of the Type 1 operation, f_i' is replaced by 1 and f_j' by $f_j' - 1$, increasing the product of the right two terms above and leaving all other terms unchanged. This yields a net increase in reduction.

Consider the Type 2 operation. The reduction of an intermediate function f' can be expressed as

$$R_{BT}(f') = \gamma(f') \sum_{l=1}^{\lfloor \frac{r-1}{\kappa} \rfloor} \left[\prod_{\substack{k=0 \\ k \neq i}}^{q-1} (r - lf_k')\right] (r - lf_i')$$

where $f_i' > 1$. An application of the Type 2 operation replaces $\gamma(f')$ by $\gamma(f') - 1$ and $(r - lf_i')$ by $(r - l(f_i' - 1))$, leaving the other terms unchanged. Since $f_j' \geq 1$ for all $0 \leq j \leq r^n - 1$ and $f_i' \geq 2$, $\gamma(f') \geq r$ and $(r - lf_i') \leq r - 2$. It follows that $(\gamma(f') - 1)(r - l(f_i' - 1)) > \gamma(f')(r - lf_i')$, and there is a net increase in reduction. Q.E.D.

Theorem 1 states that with the linear cost function, the function to add to the basis cost-table to produce a *minimal* cost-table of size 1 larger contains only 0's and 1's, However, it does not show how many of each logic value occurs. Let α_i be the number of components with logic value i for such a function, where $i \in \{0, 1\}$. Because of the symmetry of the linear cost function, the basis cost-table plus *any* function f' with α_1 1's (and α_0 0's) is a minimal cost-table. An n-variable r-valued function has $q = r^n$ components, and so $\alpha_0 = q - \alpha_1$.

$$R_{BT}(f') = (\alpha_1 - 1) \sum_{l=1}^{r-1} \left[(r)^{q-\alpha_1}(r - l)^{\alpha_1}\right],$$
$$= r^q(\alpha_1 - 1) \sum_{l=1}^{r-1} \left(\frac{r - l}{r}\right)^{\alpha_1}.$$

If we view $R_{BT}(f')$ as a continuous function of α_1, we can take the derivative with respect to α_1.

$$\frac{dR_{BT}(f')}{d\alpha_1} = r^{r^n} \sum_{l=1}^{r-1} \left[\left(\frac{r - l}{r}\right)^{\alpha_1} \cdot \left[1 + (\alpha_1 - 1)\ln\left(\frac{r - l}{r}\right)\right]\right].$$

Setting the derivative to zero, we find, in Table II, values for α_1, the number of 1's in the best function to add to the basis cost-table for one-variable r-valued functions.

TABLE II
NUMBER OF 1'S IN THE BEST FUNCTION TO ADD TO BASIS COST TABLE OF r-VALUED ONE-VARIABLE FUNCTIONS VERSUS r

r	Number of 1's	
4	3.646	4
6	4.559	5
8	5.321	5
16	7.670	8
24	9.469	9
32	10.985	11
48	13.528	14
64	15.671	16

VIII. CONCLUDING REMARKS

Our interest in the cost-table technique is inspired by its fundamental nature. Design by cost-table combines components so that the design specifications are achieved at the lowest possible cost. In logic design, the design specification and components are both logic functions. This simplification makes logic design a good first choice to study the cost-table method, and we are able to gain insights not possible with less formally specified design problems. Even within logic design, simplifying assumptions are necessary. Our analysis of heuristics for finding minimal cost-tables could not have been done on functions with two or more variables; the set of all functions is too large for a computer analysis (while the number of one-variable 4-valued functions is 256, there are 4×10^9 two-variable 4-valued functions!). However, the three key results, Lemmas 1, 2, and Theorem 1, as well as other material in Sections V through VII, apply to the general case of n-variable r-valued functions. Further, the results on one-variable 4-valued functions in Sections III and IV suggests that similar phenomena occur for the case of functions with more variables. For example, it is unlikely that, for more than one variable, the point of diminishing returns with respect to cost-table size will be exceeded. That is, for a practical number of inputs, we expect even a small increase in cost-table size to produce a large decrease in total cost. We also believe that another observation applies to the more general

case. The variation observed in the performance of heuristics suggests that it is important to find good heuristics. For example, an explanation of the surprisingly poor performance of MAXIMUM_REDUCTION requires a careful examination of this heuristic. The analysis shows that, for small sizes, the *average* cost-table performance is far from minimal.

We believe that the design of heuristics warrants further study. In spite of the existence of a reasonably good heuristic, ITERATIVE_BEST, it is difficult to find provably minimal cost-tables. We have done so for cost-tables of size 1 larger than the basis cost-table using the linear cost. We feel that a productive line of attack on this problem is to identify functions that will not be in a minimal cost-table. We have shown that composite functions are not candidates for elimination. However, other functions may be removed from consideration in large cost-tables (e.g., Lemma 1 of [13]).

We believe that the linear cost function also warrants further study. It is interesting that a minimal cost-table associated with a linear cost function such that $c_q + s > 0$ is also a minimal cost-table for any other linear cost function. We observe that linear cost functions exist in which the best function to add to the basis cost-table is one with high cost, rather than other less costly (but less advantageous) functions. Thus, low cost alone should not be a criteria for selecting cost-table functions. This statement is also supported by the observation that the heuristic MAXIMUM_REDUCTION, which tends to choose low cost functions, is a poor heuristic.

ACKNOWLEDGMENT

The authors thank the referees for constructive comments that led to improvements in this paper.

REFERENCES

[1] M. H. Abd-El Barr, "Design of multi-valued circuits for CCD and MOS implementation," Ph.D. dissertation, Univ. of Toronto, Toronto, Ont., Canada, 1986.

[2] M. H. Abd-El Barr, Z. G. Vranesic, and S. C. Zaky, "Synthesis of MVL functions for CCD implementations," in *Proc. 16th Int. Symp. Multiple-Valued Logic*, May 1986, pp. 116–127.

[3] M. H. Abd-El Barr, T. D. Hoang, and Z. G. Vranesic, "The incremental-cost approach for synthesis of CCD 4-valued unary functions," in *Proc. 18th Int. Symp. Multiple-Valued Logic*, May 1988.

[4] C. M. Allen and D. D. Givone, "A minimization technique for multiple-valued logic systems," *IEEE Trans. Comput.*, vol. C-17, pp. 182–184, Feb. 1968.

[5] J. T. Butler and K. A. Schueller, "On the equivalence of cost functions in the design of circuits by costtable," *IEEE Trans. Comput.*, vol. C-39, pp. 842–845, June 1990.

[6] H. G. Kerkhoff and M. L. Tervoert, "Multiple-valued logic charge coupled devices," *IEEE Trans. Comput.*, vol. C-30, pp. 644–652, Sept. 1981.

[7] H. G. Kerkhoff and H. A. J. Robroek, "The logic design of multiple-valued logic functions using charge-coupled devices," in *Proc. 12th Int. Symp. Multiple-Valued Logic*, Paris, France, May 1982, pp. 35–44.

[8] J.-K. Lee and J. T. Butler, "Tabular methods for the design of CCD multiple-valued logic," in *Proc. 13th Int. Symp. Multiple-Valued Logic*, Kyoto, Japan, May 1983, pp. 162–170.

[9] D. M. Miller and J. C. Muzio, "On the minimization of many-valued functions," in *Proc. 9th Int. Symp. Multiple-Valued Logic*, Bath, England, May 1979, pp. 294–299.

[10] S. Onneweer, H. G. Kerkhoff, and J. T. Butler, "Structural computer-aided design of current-mode CMOS logic circuits," in *Proc. 18th Int. Symp. Multiple-Valued Logic*, May 1988, pp. 21–30.

[11] H. A. J. Robroek, "The synthesis of MVL-CCD circuits," M.Sc. Rep. 12.3936, Twente Univ. of Technology, Enschede, The Netherlands, Dec. 1981.

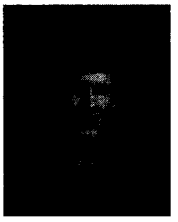
[12] K. A. Schueller, "The costtable approach to the logic design of multiple valued logic circuits," Ph.D. dissertation, Northwestern Univ., Evanston, IL, Aug. 1987.

[13] K. A. Schueller, P. P. Tirumalai, and J. T. Butler, "An analysis of the costtable approach to the design of multiple-valued circuits," in *Proc. 16th Int. Symp. Multiple-Valued Logic*, Blacksburg, VA, May 1986, pp. 42–50.

[14] K. A. Schueller and J. T. Butler, "The costtable problem is NP-complete," in *Proc. 28th Annu. Allerton Conf. Commun., Contr., and Comput.*, Oct. 1990, pp. 948–957.

[15] W. R. Smith III, "Minimization of multivalued functions," in *Computer Science and Multiple-Valued Logic*, D. C. Rine, Ed. New York: North Holland, 1977, pp. 221–261.

[16] P. P. Tirumalai, "Four-valued logic CCD programmable logic arrays," M.S. thesis, Northwestern Univ., Evanston, IL, June 1984.



Kriss A. Schueller (M'91) was born on November 18, 1959 in Oak Hill, WV. He received the B.A. degree in physics and the M.S. degree in mathematics from Youngstown State University, Youngstown, OH, in 1981 and 1982, respectively, and the Ph.D. degree in computer science from Northwestern University, Evanston, IL, in 1987.

Since 1987, he has been an Assistant Professor in the Department of Mathematical and Computer Sciences, Youngstown State University, Youngstown, OH. His research interests include multiple-valued logic, computer architecture, and operating systems.

Dr. Schueller is currently the Treasurer for the Multiple-Valued Logic Technical Committee of the IEEE Computer Society.

Jon T. Butler (S'67–M'67–SM'82–F'89) was born in December 26, 1943, in Baltimore, MD. He received the B.E.E. and M.Engr. (E.E.) degrees from Rensselaer Polytechnic Institute, Troy, NY, in 1966 and 1967, respectively, and the Ph.D. degree in electrical engineering from the Ohio State University, Columbus, OH, in 1973.

Since 1987, he has been a Professor in the Department of Electrical and Computer Engineering of the Naval Postgraduate School, Monterey, CA. From 1974 to 1987, he was on the Faculty of Northwestern University, Evanston, IL. During that time, he served two periods of leave at the Naval Postgraduate School, first as a National Research Council Senior Postdoctoral Associate (from 1980 to 1981) and second as the NAVALEX Chair Professor (from 1985 to 1987). From 1973 to 1974, he was a National Research Council Postdoctoral Associate at the Air Force Avionics Laboratory, Wright-Patterson AFB, OH, where he worked on the application of cellular automata to pattern processing problems. His research interests include multiple-valued logic and reliable multiprocessing systems.

Dr. Butler was Chairman of the 1980 International Symposium on Multiple-Valued Logic and was the first Chairman of the Multiple-Valued Logic Technical Committee of the IEEE Computer Society from 1980 to 1981. He was a Co-Guest Editor of a special issue of the IEEE TRANSACTIONS ON COMPUTERS and a special issue of COMPUTER both on multiple-valued logic. He has served as an Editor of the IEEE TRANSACTIONS ON COMPUTERS from 1982 to 1986 and of the Computer Society Press from 1986 to 1990. Currently, he is the Editor-in-Chief of COMPUTER, having served as Editor from 1988 to 1989 and Associate Technical Editor from 1989 to 1990. From 1986 to 1988, he served as Vice-Chair for Hardware of IEEE Computer Society's Technical Activities Board. From 1982 to 1985, he was a Distinguished Visitor of the IEEE Computer Society. He is currently a member of the Board of Governors of the IEEE Computer Society. He is the co-recipient of the Award of Excellence and the Outstanding Contributed Paper Award for papers presented at the International Symposium on Multiple-Valued Logic in 1985 and 1986, respectively.